Dr. Dai, Dr. Hero,

Attached is a recently accepted paper (PerCom 2012) authored by me and my team lead, Mr. Andrew Toth, under our project on Quality of Information (QoI). When we talk of QoI, we are referring to methods of measuring the usefulness of data/information to its user. In this paper we build a case for tracking the functionality by which the data arrives to the user, including issues such as: capturing information from the real world, processing the data, and delivery of the data. From what I understand of your research from the kickoff meeting in Ann Arbor last September(?), is that VoI should be able to help us characterize these functionalities. Thank you for your continued collaboration with ARL. I look forward to seeing further results of your efforts.

Regards,

Trevor Cook

To do: will send an electronic copy.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **2012**	2. REPORT TYPE	3. DATES COVERED **00-00-2012 to 00-00-2012**
4. TITLE AND SUBTITLE **Information Processing And Quality Evaluations**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U. S. Army Research Laboratory,Adelphi,MD,20783**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
To be presented at the IEEE PerCom (Persuasive Computing and Communications) Conference March 19-23,2012 Lugano, Switzerland

14. ABSTRACT
We examine the concept of Quality of Information (Qol); the idea that networked data systems can better support users if they have a notion of what the data is being used for. We argue that to judge quality it is as important to know how the data was processed as it is to know what the data says. We model Qol systems as a structure that enforces the annotation of data creating processes with the data they produce. Furthermore, we model the evaluation of quality as functions that map information into a closed and bounded, totally ordered set. We define addition and multiplication operations which make a subring out of the set, thus allowing data composed of multiple pieces of information to be evaluated as sums and products of its components. Finally we give an example to show how our model supports the delivery of high quality information.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**9**	

Information Processing and Quality Evaluations

Trevor Cook
U.S. Army Research Laboratory
Adelphi, MD, 20783
Email: trevor.j.cook@us.army.mil

Andrew Toth
U.S. Army Research Laboratory
Adelphi, MD, 20783
Email: andrew.j.toth@us.army.mil

***Abstract*—We examine the concept of Quality of Information (QoI); the idea that networked data systems can better support users if they have a notion of what the data is being used for. We argue that to judge quality it is as important to know how the data was processed as it is to know what the data says. We model QoI systems as a structure that enforces the annotation of data creating processes with the data they produce. Furthermore, we model the evaluation of quality as functions that map information into a closed and bounded, totally ordered set. We define addition and multiplication operations which make a subring out of the set, thus allowing data composed of multiple pieces of information to be evaluated as sums and products of its components. Finally, we give an example to show how our model supports the delivery of high quality information.**

I. Introduction

The purpose of data and communication networks is to deliver information to their users. Since their inception, we have been improving the performance of data networks. Most of the gains have been in the raw power of networks to deliver data. That is, we have made tremendous gains in the speed and reliability of data delivery. However, there is an often overlooked aspect of data delivery, and that is how well it supports the tasks users are performing with the delivered data.

It is shown in [1] that performance of tactical military networks are intricately linked to both the information that they provide and to the social networks which they support; and that treating them independently is insufficient for tactical settings. Using several motivating examples centered around intelligence gathering for counterinsurgency, the authors note that the need for networks to process and deliver information in these situations far exceeds their ability. They conclude that if networks are enabled with the ability to evaluate the quality of information (QoI) that they are delivering, they will better serve the tasks for which they are being used.

In [2], the authors show the need for some notion of QoI in sensor networks. They define QoI as "The collective effect of information characteristics (or attributes) that determine the degree by which the information is (or perceived to be) fit-to-use for a purpose." This notion is refined in [3] to be split between QoI–inherent attributes of quality–and value of information (VoI)–fitness for use.

There are many ways in which inherent attributes of information quality may be assessed, for instance "Correctness" and "Freshness," are some of the QoI metrics described in [1], [2]. Attributes such as these are also termed data quality (DQ) in [4], or information quality (IQ) [5]. These all represent different factors of data that contribute their usefulness.

A. Quality as Preference

Although intrinsic qualities are important to deciding the fitness of use, in this paper we are concerned with QoI in the sense of [1]; that is, how well a piece of information will support a task. We want to enable information systems to deliver the information that the consumer will find the most useful. To do this, the system must be able to compare any two pieces of information and decide which one is better (or if they are of equivalent value). In other words, we define QoI as a total ordering of information.

QoI as a total ordering of information can be thought of as a function which maps information to an ordered set; such as the interval [0,1]. Since each user will have a different preference over the information, each user may have a different function which maps data to (for instance) [0,1]. The aim of this paper is to provide a foundation for such information preference functions.

The foundation we propose focuses on information processing as a key consideration in determining preference. A QoI system should therefore keep track of data processing so that fully informed quality evaluations can be made. We begin this discussion on how this can be accomplished with an overview of the life cycle of a piece of information, from creation to its evaluation of quality.

B. Information Processing and Quality Evaluation

Fig. 1 models the information generation and evaluation process. Keeping the definitions of data and information vague for now; this diagram represents the view of any individual information system which can generate or receive data, process it, and evaluate its quality. In this diagram and the others presented in this paper, circles represent some type of object–be it a data source entity, a piece of data, or a numerical value–and arrows represent transformations from one object to another. Reading the diagram from left to right, we get a rough explanation of the process of generating and processing data and then rating its quality.

The left most circle, "Source," represents any entity or phenomenon from which data can be brought into the system. The arrow labeled "Capture" represents the functionality by which we bring the data into the system. Capture encompasses actions like taking a digital image, checking the current value

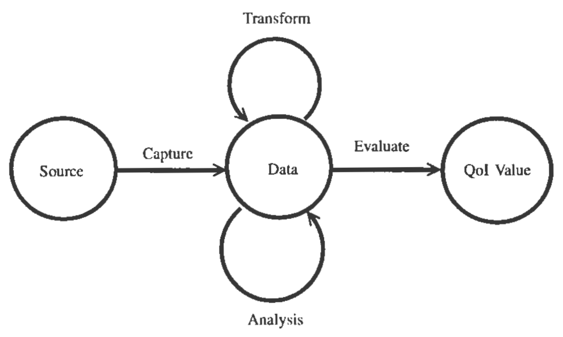

Figure 1. The top level view of a QoI system.

of a timer, or receiving data from another system. The sources in these cases then are some physical location, time, and the remote system, respectively.

Once data is in the system, we define two data processing operations that result in data, "Transform" and "Analysis." Transforms represent operations in which data is changed into a new form; such as in image compression. Analyses represent operations which look at a piece of data to derive information. An example of an analysis could be an image recognition software that tags images with with the names of recognized objects. Analysis and Transform are self arrows; they begin and end with data, and so we can compose any number of them in succession.

At any point in the succession of analysis and transformation we can evaluate data's quality. This is represented by the "Evaluate" arrow, which maps information into a specific numerical domain; represented with "QoI Value". As discussed above, the QoI Value domain allows us to rank the relative merits of pieces of data. There may be many appropriate domains for measuring value, however, in this treatment all QoI Value domains must at least constitute a totally ordered set. As in the case of the other arrows in Fig. 1, there are numerous possible ways to satisfy the Evaluation functionality. Each possible evaluation function is a particular way of measuring the quality of information, and so evaluation functions are the means by which users express their preferences over the data.

Fig. 1 serves as an introduction to our view of the relevant features which must be accounted for in evaluations of information quality. It is a starting point from which we will make departures in the rest of this paper. In Section III we develop a more detailed model of the Capture, Transform, and Analysis functions. In Section IV we develop a particular quality domain and discuss the details of evaluating quality, and in Section V we give an example of how a QoI system may processes an information request. Before information is processed and evaluated however, we first define what we mean by information.

II. Data and Information

In this section we look at the thing we are ordering; information. An ordering may reflect a number of underlying attributes of information, for instance completeness or freshness. However, as described in [4], [5], and others, we suppose that preferences generally reflect the information's suitability for use in one way or another. For a complete definition, we therefore wish to include all the relevant features that will help a user decide how useful the information will be.

With the aim of capturing features that indicate suitability of use, simple values–i.e. numbers–do not make a good basis for a definition of information since they lack context. For instance, "1429" may represent any number of things, but "1429 UTC" (Universal Coordinated Time) is fairly specific. So as a starting point, we find typed values to be more useful than untyped values.

For the sake of quality, we find that simple typed values are not enough. For example, if we have a piece of data containing a UTC time stamp, we still can not reliably say how recent is the data. However, if we knew that the time stamp was generated by a reliable GPS unit and stored without being amended, then we can be fairly certain of the data's age. For this reason, we specify that information is a function-value pair, $I = (f, V)$, with f the function which generated the typed value, V. This view is supported in [6], which describes how military analysts evaluate the quality of a piece of data. Knowledge of how the data was generated (i.e. function) is one of the two major issues that they evaluate. The other is of course the content of the data (i.e. value).

Describing a piece of information as a function-value pair misses some relevant context since it stops short of a description of the source. However, in many cases, the source is implied by the function. For instance, "wget" is a Gnu utility (function) for non-interactive web downloads, and the function call: "`wget http://host.com/file.pdf`," is pretty explicit.

Another example of capture functions with an implicit source is a camera, whose source is the physical world. It is often useful to know where in the world an image is captured from. However, a specific description of the image source, i.e. an exact location of where the image was taken, requires additional functionality such as GPS. Therefore, knowledge about sources can be thought of as additional pieces of information, (f, V). This takes into account that our knowledge of sources is dependent on some functionality.

For the reasons above, we find function-value pairs to be a minimum complete representation of information from which we can judge quality. The above example of indicating an image's source with a piece of GPS information underlines the need to join pieces of information together. This leads to our definition of data.

We define a piece of data, D, as a set of information. That is, $D = \{I_i\} = \{(f_i, V_i)\}$, for some finite set of indexing values, i. We arrived at this definition because it greatly simplifies the view of functions on data. Specifically, it allows us to look at data processing and quality evaluation as built from the composition of atomic operations on individual pieces of information. However, we will also consider data to be a type of value. In this way, data can also be thought of information that lacks functional context.

This view of data and information contrasts with a traditional view that data is in some way "raw" and information is "processed." We do not find this traditional contrast of data and information to be helpful because it implies that the processed information is more fit for use. Describing a method by which we can determine data's or information's fitness for use is the purpose of this paper, and so we cannot have it be assumed in our underlying definitions.

Viewing data as merely a *set* of information contrasts other common views which assume that data is structured or that they have representative types. For instance, image files are primarily images, though they often contain secondary meta-data such as the date, time, GPS location, and the type of camera that took the picture.

Data structure is preserved somewhat through the use of typed values, which let us distinguish between different types of data. From our point of view, the actual structure is not important, however. Only the fact that the data contains elements of information and that the system is aware how to access those elements is important.

Describing data as a set of information has an important consequence in that we do not differentiate between data and meta-data. Instead, everything we consider is just information and collections of information; over which the same kind of processing and evaluations apply.

III. Information Processing

In this section we discuss the specifics of the functionalities represented by arrows ending in data in Fig. 1; namely Capture, Transform, and Analysis. This model is useful because it outlines commonalities of structure and gives us a few simple methods by which we can enforce the processing of *information* (from which we can fully evaluate quality), as opposed to processing *data* (which gives a partial view of quality).

A QoI system is responsible for keeping track of which processes resulted in which data; storing both together as what we refer to as information. It would be impractical for every data process to perform this task for its own data. Moreover, it may be impossible since we shouldn't assume that the process is truthful. Instead, we show how a system can perform this task using a few functionalities that take processes as arguments and return information yielded by applying these processes.

As an additional note, in the rest of this paper, we hold to the convention that the functionality from Fig. 1 will be referred to as a proper name, e.g. Capture, and that other functions are designated a lowercase letter, e.g. c, t, f. Values are designated with upper case letters, e.g. S, V, D. We indicate the possibility of the existence of multiple elements in a set with subscript, e.g. $\{(f_i, V_i)\}$, but generally disregard the indexing set ($i \in \{1, ..., N\}$) because it is unimportant to the discussion.

A. Capture Functions

Capture functions are the means by which data is admitted into an information system. As shown in Fig. 1, these functions operate on (data) sources to produce data.

Many things may be considered as capture functions. For example, a camera is a function and the world itself is the source on which it operates. Other examples of capture functions are: timers–which create data based on time; GPS–which create data based on physical location; and downloads–which create data based on data from other systems.

It may seem strange to think of ethereal things such as time to be data sources. However, timers are generally some form of counters connected to materials that oscillate as functions of time. So, the source of the functions should be thought of as time itself.

As discussed in Section II, in order to gain some insight into the utility of the data it is important to know how it was generated. For instance, the capture function which is a laptop clock has quite different characteristics than the capture function which is a GPS clock–even though both methods are functions of time and may yield the same type of value. The difference is that one tends to reflect the truth of the source better than the other.

We build the Capture functionality of Fig. 1 from atomic capture functions, c. If S is some data source, c, is defined as a function which operates on S to produce some type of value, V:

$$c : S \to V.$$

The Capture functionality is built by using an annotating application function, run;

$$run : c, S \to \{(c, c(S))\},$$

which takes c and S as arguments and returns a piece of data consisting of a single a piece of information. The information is created by applying c to S, and recording c as the generating function.

The difference between a capture function, c, and the "Capture" functionality is that c represents cameras, timers, and the like, which produce a single value with no notion of QoI. On the other hand, "Capture" is the QoI system functionality that keeps track of what data came from what function.

Time, location, and digital images are valuable pieces of data in and of themselves, but it is often the case where information is associated with each other. This can be seen in standards such as Exif and JPEG 2000; which store alongside images additional information in the form of meta-data tags.

To account for bundling of information, we include the ability to add pieces of information together; as shown in Fig. 2. In this figure, existing information $\{(f_i, V_i)\}$ is combined with captured information $\{(c, V)\}$ at the top of the diagram. Next, the *concat* function joins $\{(f_i, V_i)\}$ and $\{(c, V)\}$ into a single set, shown by the return to the central $\{(f_i, V_i)\}$ object. In this way, we can represent complex data as a series of atomic captures concatenated together. For instance, we can represent things such as the Exif photo with meta-data; with the image, time-stamp, location, and other pieces of meta-data accounted for as individual capture functions.

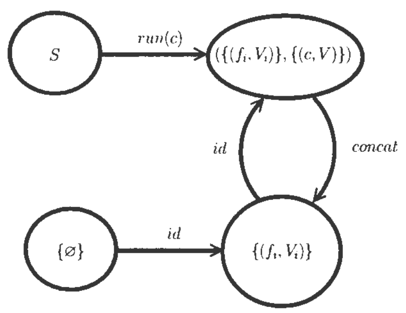

Figure 2. Generating data through capture functions as viewed as repeated captures added to an initial piece of null data.

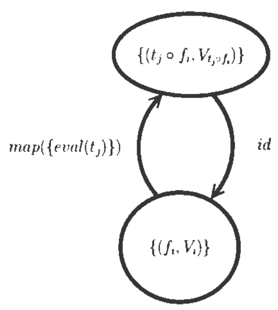

Figure 3. Transform functionality represented as an application of t_j to each piece of information in the data.

In keeping with the structure agnostic view of data, we have chosen $concat$ to be a simple joining of sets. However, this could be extended to incorporate structure. To do this, we could define $concat$ to accept a function h, which would enforce a hierarchy on the joined elements. In this case, the arrow in the diagram would be labeled with $concat(h)$ instead of $concat$.

Another interesting feature of Fig. 2 is the inclusion of the null data, $\{\varnothing\}$; which can be viewed as data with zero information. We connect $\{\varnothing\}$ to $\{(f_i, V_i)\}$ with the identity function, id, to show that $\{\varnothing\}$ is data. Including $\{\varnothing\}$ is essential in this figure; it acts as the initial piece of data to which all future information is added.

B. Transformation Functions

Data transformation is a type of data processing wherein the information within a piece of data is changed. This can embody processing operations such as compression, translation, or deletion of information from a piece of data. Like Capture, the "Transform" functionality of Fig. 1 is built from atomic transformation functions.

Atomic transformation functions, t, are defined as functions that operate on typed values:

$$t : V \to V,$$

with the two V not necessarily of the same type. Examples of t are a routine that compresses an image, and one that converts a timestamp to a different format. The Transform functionality is shown in Fig. 3 with the arrow labeled $map\left(\{eval\left(t_j\right)\}\right)$, and works by applying a set of transforms, t_j, to each piece of information in the data $\{(f_i, V_i)\}$.

As was the case of Capture's "run," transformation uses a helper function, $eval$, of the form

$$eval : t, (f, V) \to \left(t \circ f, t\left(V\right)\right),$$

which applies the transform to the value, and records the transform as a function composition. The other helper function, map, of the type:

$$map : \{f_i\}, \{X_j\} \to \{f_i(X_j)\},$$

takes a set of functions and applies them to a set of arguments so that every function gets applied to every argument.

Transforms are defined so that each atomic transform function is applied to each piece of information in the data. This is necessary for unstructured collections. However, for this to work we must allow the transform functions to determine whether or not they can transform a value. If value V_i is inapplicable to transform t_j, then t_j must return a null value, $\varnothing$. Subsequently, $eval$ must also support the failure of transform application.

We define

$$eval : t, (f, V) \to \varnothing,$$

when $t\left(V\right) = \varnothing$. The Transform functionality therefore will only include information from valid transforms, and so we can ensure every piece of information is only transformed one way (if we want to) by only including one transform for every type of value in the data.

Allowing individual transforms to fail is more than just a way to ensure the Transform functionality behaves well for unstructured sets. It is fundamentally more sound for the function to determine applicability than for the system. For if it was the system's responsibility, the system would still have to know details of the transform, t. In other words, how the system behaved would be function of t regardless. However it need not be a function of the system as and so it isn't.

C. Analysis Functions

Data analysis is the act of deriving information from data. Analysis represents processes such as: counting the number of words in a document; calculating the PSNR of a compressed image; tagging an image; or performing a speech to text transcription. The important feature in Analysis, as we define it, is that the result of the analysis is added to the analyzed data. As in the case of Capture and Transform functionalities, the Analysis functionality shown in Fig. 1 is built from atomic functions, a:

$$a : D \to V,$$

which yields a typed value based on a piece of data.

The Analysis functionality is shown in Fig. 4. The analysis is performed by the arrow labeled $\{id, run(a)\}$; which represents a function that returns a set containing a copy of the data, $id\left(\{(f_i, V_i)\}\right)$, and an analysis of the data, $run(a, \{(f_i, V_i)\})$.

As in the case of transforms, we account for the fact that the analysis may be unsuitable for the data by allowing it to return

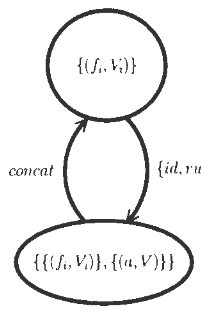

Figure 4. Analysis functionality is represented as an analysis, a, applied to the data and resulting in a piece of information, (a, V), which is then concatenated to a copy of the data.

the null value, $\varnothing$. The analysis is recombined with the original data through the *concat* function as in the case of Capture. The result is just a new piece of data of exactly the same form as the original piece of data. So, we can perform as many analyses as necessary, deriving more and more information from the data, but still ending up with a piece of data.

In [4], the authors show many different ways to quantify quality, e.g. Accessibility, Believability, Understandability, and many others. For such complex notions, we don't presuppose that there will be any one canonical metric for any given type of data, so in our model such metrics would be considered particular analyses. Similarly, analysis accounts for inherent quality attributes in the QoI level of the split view of [3].

As a final note, when we consider data, D, to be a type of value, a file system becomes a piece of data of the form: $D_{filesystem} = \{(f_{QoIsys}, D_{file})\}$. Seen in this regard, information fusion can be represented through the use of transforms to select subsets of files followed by an analysis which creates a new file based on the subset.

D. Network considerations

In our model, quality is a subjective measurement of data's usefulness to the requester of the data. Information can be transmitted to the consumer over sometimes unreliable or constrained networks, and the delivery will affect the quality. For instance, it is easy to see that bit errors affect quality, but the timeliness of delivery can affect the data's usefulness as well.

We can allow a QoI system to take into account considerations of data delivery through the use of the processing operations already discussed thus far. This includes aspects such as packaging the data, capturing the network state, predicting (analyzing) the effect of the network on transported data, and modeling the transport as a capture function.

We will give an example of these considerations in Section V. At this point, it is important to note that quality is ultimately judged by the consumer, but the effect of the transport must be accounted for by the server before the process of sending is performed.

IV. Quality Evaluations

A. Quality Mathematics

Having developed our view of information–its form and processing–we begin describing a framework for evaluating the quality of information. As in the case of the processing functionalities, we wish to build data quality evaluations from individual information evaluations. Our basis for evaluating quality is a function, q, of the type:

$$q : (f, V) \rightarrow [0, 1],$$

such that q assigns each piece of information a real number in the range [0,1].

We choose the range [0,1] because it is a totally ordered set, but also because it is a closed and bounded space on which we can define suitable addition and multiplication operations. We desire our quality metric to be bounded; that is, for any piece of information there is a maximum quality. We also want the maximum quality to be achievable (closed space). Otherwise, for any given quality we would always be able to define another one that is closer to the maximum quality.

Having a closed and bounded interval over which to measure quality is important for the implementation of quality functions. Without bounds, two quality functions may give wildly different measures for similar pieces of information. By bounding the space we help ensure that all metrics are at least on the same scale. With an open interval we have the same problem of scaling as in an unbounded function. That is, for any measure we can always define another measure which yields greater values. Closing and bounding the quality space does not ensure that all functions will create fair comparisons between information, however it alleviates some problems.

We want a method by which data can be evaluated through the composition of its information quality. To allow this we define the following two operations that allow us to add and multiply quality values together. The addition operation, $\oplus$, is defined as

$$a \oplus b = a + b - \sqrt{ab}, \qquad (1)$$

and multiplication, $\otimes$, is defined in the usual way

$$a \otimes b = ab. \qquad (2)$$

This definition of addition and multiplication constitutes a commutative sub-ring. That is, the individual operations are algebraically closed, commutative, associative, and the distributive property holds. The two identities are 0 and 1, but there are no inverses in either operation. Being a sub-ring allows us a add and multiply quality evaluations of individual pieces of information while ensuring that the resulting sums and products are valid values in the range $[0, 1]$.

B. Information Quality and Data Quality

The atomic quality function $q(f, V)$, can either make its evaluation based solely on f, solely on V, or use both f and V. Evaluations based on the generating function, f, allow us to prefer certain pathways of information over others. For instance, to prefer GPS based time over computer clock based time, or to prefer certain types of analysis over others.

Evaluations based on the value, V, are preferences over what the information says. In other words, the relevance of the data to the request. We see examples of these type functions in data base and URL queries. Evaluations based on a both f and V, may be useful when different functions return similar values and both pathway and relevance must be weighed.

We wish to evaluate data through evaluations on the component information. Using the quality space and (1) and (2) defined above makes this easy. Since the definition constitutes a subring, all data evaluations built from sums and products of constituent information will resolve to a valid evaluation in the quality domain $[0,1]$.

For example, if $D = \{I_1, I_2\}$ and $\hat{q} = \{q_1, q_2\}$ then we can map all $q_i \in \hat{q}$ to all $I_j \in D$ in a manner similar to how transforms are mapped to pieces of information in data. In matrix form, we create a system of quality evaluations:

$$\hat{q}(D) = \begin{bmatrix} q_1(I_1) & q_1(I_2) \\ q_2(I_1) & q_2(I_2) \end{bmatrix}. \qquad (3)$$

The abuse of notation, $\hat{q}(D)$, should be taken to mean a mapping of the elements of $\hat{q}$ over the information in D.

Next, we calculate the total quality, Q, by applying a linear weighting of the relative importance of the different quality measurements. That is, using matrix multiplication–with elements summed and multiplied with $\oplus$ and $\otimes$ instead of the usual addition and multiplication–$Q = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \hat{q}(D) \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, with $0 \leq w_i \leq 1$.

Speaking generally about quality evaluations, we have multiple ways to rate data, $\{q_i\}$, but don't necessarily have a clear cut way of comparing any two q_i. So the evaluation of Q constitutes a multi-objective optimization over the q_i objective functions. This should be expected, since there are many attributes of data consumption that add to a user's experience.

V. Example

A. Scenario

An Army commander planning a mission needs to know whether or not a seasonal road is still passable after the heavy rains of two days prior. Data related to this information need is contained in several places: a recent satellite image of the region; a mission plan stating a squad in a different company intended to use the road; and a forecast of general terrain conditions as part of a command weather report.

The goal for the above scenario is for the commander to query the network for information of type "route condition", and to receive in return recent and reliable "route condition" information derived from one of the above data. For this scenario the freshness of the data is a consideration, since information from before the rain is useless and the more recent information will more accurately reflect the current state of the roads. Other considerations are the precision and accuracy of the information–only information on a specific pass is needed and bad information could lead to a waste of resources and endangerment of life. Finally, a go/no-go decision has to be made in 12 hours, and so the request has some time for distribution and response.

B. Quality Function

The commander creates a request for information which gets translated into a quality evaluation function:

$$q_{req}(f, D) = \hat{q}_{time}(D)\left(\hat{q}_1(D) + \ldots + \hat{q}_N(D)\right). \qquad (4)$$

(The less cumbersome addition and multiplication symbols are used here, but should be taken to mean (1) and (2).) The equation specifies the total quality, q_{req}, to be the product of a response timeliness quality, $\hat{q}_{time}$, and a summation of the qualities, $\hat{q}_i$, of N known ways to derive "route condition" information from data. The hatted functions here, $\hat{q}$, are a further space saving abuse of the notation from (3), and should be taken to mean $[1,\ldots,1]\,\hat{q}(D)\,[1,\ldots,1]^T$ in the former context. In this treatment, unhatted q are used to evaluate the quality of a piece of information while $\hat{q}$ are used to look for the sum total of information evaluations within a value.

This evaluation function, q_{req}, works on the output of a capture function that prepares data for evaluation, c_{prep}. This function is responsible for applying appropriate data processing on existing data and is part of a QoI optimization process that searches through various processing options available to the system. Thus, q_{req}, evaluates various preparations of data.

On the evaluation of a particular data preparation, $q_{req}(c_{prep}, D_{prep})$, the timeliness term, $\hat{q}_{time}(D_{prep})$, evaluates nonzero results only when D_{prep} contains an analysis, (a_{ToA}, V_{ToA}), which estimates time of arrival. Specifically, $\hat{q}_{time}$ evaluates to the probability that the 12 hour deadline will be met or to zero when the analysis is not performed. The analysis, a_{ToA}, operates on data which contains information pertaining to how a particular piece of data will be sent, (c_{send}, D), and to the network state, (c_{net}, V_{net}). Therefore $\hat{q}_{time}$, and by extension (4), is nonzero only if $\{(c_{send}, D), (c_{net}, V_{net}), (a_{ToA}, V_{ToA})\} \subseteq D_{prep}$.

Eq. (4) will also only be nonzero if at least one content evaluation, $\hat{q}_i$, is nonzero. Each $\hat{q}_i$ requires data to be actually sent, and so D_{prep} must contain a piece of information such as (c_{send}, D). Sending data is modeled as a capture, since it can be seen as a capture from the requesters point of view. For all $\hat{q}_i$, the data being sent must also contain some piece of information with a value of type "route condition". However, no evaluation is made based on the content of the value. The evaluation function doesn't care if the route condition is "passable", or "flooded".

C. Quality Evaluation

The request, q_{req}, first reaches the data system at the outpost from which the commander is operating. Since the system is local and delivery has little delay, all properly prepared data yields a nonzero timeliness evaluation, $\hat{q}_{time}(D_{prep})$. The only data on this system for which the content evaluation is nonzero are preparations of the weather report: as evaluated by $\hat{q}_1 = \{q_{weath}\}$, with:

$$q_{weath}(f, D) = q_{send}(f)\,\hat{q}_{fresh}(D)\left(\hat{q}_{opt1}(D) + \hat{q}_{opt2}(D)\right).$$

The prepared data may evaluate to a positive number if it contains: a piece of information for which the function is c_{send} (as evaluated by $q_{send}(f)$ and which may prefer different methods of delivery); and whose value (D) has a piece of information indicating a recent timestamp (using $\hat{q}_{fresh}(D)$ which gives greater values to the more recent timestamps), and whose value has one of two content options ($\hat{q}_{opt}(D)$). The options reward either containing a weather report and an analysis that highlights the paragraphs containing "route condition" or sending just the paragraphs containing "route condition". Therefore the preparation $\{(c_{send}, \{(t_{keepA} \circ a_{find}, V_p), (t_{keepB} \circ c_{time}, V_t)\})\}$, derived from sending a transform of weather data which only kept the analysis and timestamp from the original weather report, would yield a non-negative quality evaluation.

The QoI system replies to the commander the most suitable preparation of the weather data, which happens to be the full report with highlighting. The request also makes its way via satellite to a nearby forward operating base. The base data system contains recent satellite images of the route. As with the weather report, there are analyses available to determine route conditions from the image. Due to limited availability of the link however, the timeliness constraint dictates that sending back the result of the analysis is the only preparation of information with positive quality. The commander prefers a full image to the analysis, but he prefers the analysis to a late image.

We have assumed that there are analyses in the system that will result in the expected data type. However, we make no stipulation that the analyses have to be automated. For example, the request by the commander could have prompted a human analyst to look at the imagery and determine the route conditions. This opens the possibility to define multiple human intelligence based functionalities, and allows the system to send back non-"route condition" data if it is aware of human analysis that transforms appropriate imagery into "route condition".

This notion can be extended further to to account for other ways of bringing information to the user. For example, it was stated that a squad leader filed a mission plan which showed an intention to use the route. One might infer then that putting the requester in touch with the squad leader may deliver high quality information. In other words, the requester will have the desired information by performing a capture, e.g., $run\,(c_{talk} \circ c_{SgtJones})$–talk to Sgt. Jones. In this case, the system is not sending the information per se but will instead be initiating a capture (presumably by sending a message).

VI. Conclusion

In this paper we discussed various aspects of evaluating information quality. Primarily, we highlight the importance of data processing operations on data quality. We indicate that a quality aware system should have responsibility for tracking the data processing and associating the processing with the data that it creates. We show how this can be implemented with a few basic patterns and operations such as *map*, *concat*, and *eval*.

Next, we described a suitable mathematical framework for the evaluation of quality values. We presented reasoning that describes how evaluations of quality should be normalized to the range $[0, 1]$, and show how the quality of a piece of data can be composed through the addition and multiplication of evaluations of component information.

Finally, we gave an example in which a military commander requested information on the conditions of a particular route, how that information might be derived from analyses on various types of data, and how quality functions might evaluate these derivations. Furthermore, we postulated that human analytical and information gathering capabilities might be accounted for and utilized by a QoI system.

In our example, the commander's request related to a specific type of data. This point is important because it allows the QoI system to determine how to derive needed information. Specifically, by modeling the different data types as vertices, functions with explicit input and output data types become edges. Preparations of data are thus some path between existing data and requested type.

This indicates a way forward for the design of an actual QoI system. Classes of information requests should be identified and classified as various data types. Next, system functionality must be identified in terms of the types they require and provide. Having this, a system can satisfy requests by finding paths to required type from known (or capturable) types. These are of course not easy tasks (and we haven't even mentioned efficient ways of finding high valued paths nor methods for creating well balanced quality functions). The hard problems are still to be solved. Hopefully, however, this analysis gives some insight into where the effort should be spent.

References

[1] A. Bar-Noy, G. Cirincione, R. Govindan, S. Krishnamurthy, T.F. LaPorta, P. Mohapatra, M. Neely, and A. Yener, "Quality-of-information aware networking for tactical military networks", in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, march 2011, pp. 2 –7.

[2] C. Bisdikian, J. Branch, K.K. Leung, and R.I. Young, "A letter soup for the quality of information in sensor networks", in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, march 2009, pp. 1 –6.

[3] C. Bisdikian, L.M. Kaplan, M.B. Srivastava, D.J. Thornley, D. Verma, and R.I. Young, "Building principles for a quality of information specification for sensor information", in *Information Fusion, 2009. FUSION '09. 12th International Conference on*, july 2009, pp. 1370 –1377.

[4] Richard Y. Wang and Diane M. Strong, "Beyond accuracy: what data quality means to data consumers", *J. Manage. Inf. Syst.*, vol. 12, pp. 5–33, March 1996.

[5] Besiki Stvilia, Les Gasser, Michael B. Twidale, and Linda C. Smith, "A framework for information quality assessment", *Journal of the American Society for Information Science and Technology*, vol. 58, no. 12, pp. 1720–1733, 2007.

[6] Hongwei Zhu and Richard Y. Wang, "Information quality framework for verifiable intelligence products", in *Data Engineering*, Yupo Chan, John R. Talburt, and Terry M. M. Talley, Eds., vol. 132 of *International Series in Operations Research & Management Science*, pp. 315–333. Springer US, 2010.